ELSEVIER

Journal of Computational and Applied Mathematics 63 (1995) 465–473

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

Model of neurocontrol of redundant systems[1]

Alexander Frolov[a], Stanislav Řízek[b,*]

[a] *Institute of the Higher Nervous Activity and Neurophysiology, Russian Academy of Sciences, Butlerova 5a, 117865 Moscow, Russia*

[b] *Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic*

Received 25 November 1994; revised 20 April 1995

Abstract

Design of a controller generally requires generation of the inverse transfer function of the plant. In the case of control by velocity it involves calculation of pseudoinversion of Jacobian of the plant. This process may be rather complicated in systems with many parameters. A new method of differential control of redundant systems has been developed. It is based on the analogy with the brain function in visual–motor coordination. The proposed scheme simulates both learning and operation procedures in living creatures. The method does not require calculation of matrix inverses. The model of pseudoinversion of Jacobian is generated as a layered neural network involving a hidden layer with higher-order neurons. It is being developed during the learning process based on the error back-propagation algorithm. Its convergence is analyzed for a linear case. The procedure uses differential control, local linearization, and control error estimation by a special neural network taught by Hebb's rule. It was tested on a model of a multijoint manipulator.

Keywords: Differential control; Neurocontrol; Back-propagation learning

1. Introduction

Artificial neural networks are now widely used for control of various complex nonlinear systems [5, 6, 9]. They promise a good solution for most of the control problems. Learning ability is one of their main advantages, and special learning algorithms provide rather good convergence. They do not require precise initial mathematical models that can be developed during the adaptation process. Generalization properties ensure solving situations that have not been trained in the learning phase. Physical implementations of neural networks offer the advantage of a massive

* Corresponding author. E-mail: rizek@uivt.cas.cz.

[1] This paper was supported by the Grant Agency of the Czech Republic through the Grant No. 102/93/0912 and Russian Foundation of Fundamental Research, project No. 94-04-11318.

SSDI 0377-0427(95)00071-2

parallelism, that provides a high speed of information processing under the possible low speed function of single processing elements in the operation phase. It may also provide a significant fault tolerance since the damage of several processing elements may not significantly impair the performance of the whole neural network.

Another general aspect of the belief in a neural network approach is the possibility of continuously adding the functional knowledge from neural systems of living creatures to the design of artificial neural networks. Recent neurophysiological experiments in visual–motor coordination have demonstrated that the activation of some cortical neurons considerably precedes the activation of motor neurons. It can be assumed that the plan of movement is generated in special parts of the nervous system and the reference trajectory is formed in advance [1]. Further, it has been found that some neurons of the vision system are sensitive to motion in specific directions that are called preference directions. It looks like the three-dimensional coordinate system is transformed by some neural network into an n-dimensional system of preference directions [3]. Moreover, the visual system is more sensitive to motion velocity than to object location that is especially convenient for differential control. Living creatures typically learn to coordinate their motoric systems first by generating small motions and analyzing the responses. Then, they learn more complicated movements by trying to perform them and minimizing the errors [4].

Standard control techniques usually require first the development of a mathematical model of the plant or its inverse [2]. However, the plant may not be available for complete analysis in some cases. Then the neural approach can aid in solving the problem.

The recent neurophysiological knowledge has been used in the design of a new version of a differential neurocontroller [8]. Its basic blocks resemble individual parts of biological nervous systems and the proposed scheme simulates both the learning and the operating procedures of living creatures. The proposed scheme involves two neural control blocks with higher-order neurons that are adapted in the learning process. It has been designed for control of mechanical systems by velocity, though it can be generalized for differential control of arbitrary systems.

2. Scheme of the neurocontroller

The proposed scheme of the neurocontroller is shown in Fig. 1. It consists of the following blocks: F — feedforward control block, B — feedback control block, R — movement planner, I — integrator, P — plant.

Here x^{a}, $x^{c}(t)$ and $x^{d}(t)$ are the target, current and desired output vectors of the plant; $\dot{x}^{d}(t)$ is the desired output vector derivative; $c(t)$ is the control vector and $\dot{c}(t)$ is its derivative; $u_{B}(t)$ and $u_{F}(t)$ are the partial control vectors generated by the control blocks B and F; ee is the estimated control error; t is the time parameter.

The variables are processed according to the relations:

$$x^{c} = P(c), \tag{1}$$

$$u_{B} = B(c)\cdot x^{\Delta}, \tag{2}$$

$$u_{F} = F(c)\cdot \dot{x}^{d}, \tag{3}$$

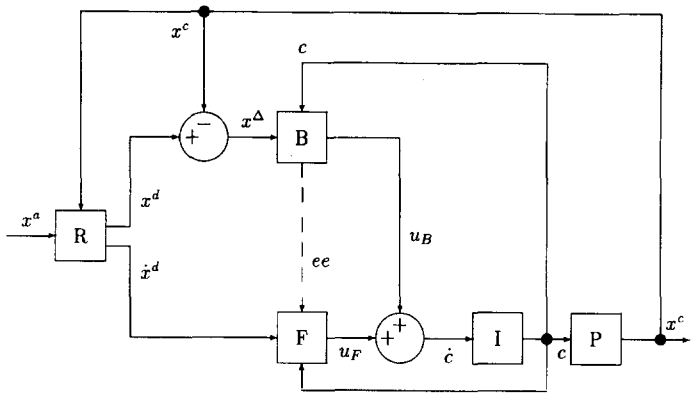

Fig. 1. Scheme of the differential neurocontroller.

$$\dot{c} = u_B + u_F, \tag{4}$$

$$c = \int \dot{c}\,dt, \tag{5}$$

where $x^\Delta = x^d - x^c$ is the discrepancy between the desired and current output vectors.

Generally, the plant should be treated as a dynamic system and described by the system of differential equations in the case of continuous time, or by the system of difference equations in the case of discrete time. However, in many problems the characteristic times of the desired changes of the system state are much larger than the time constants of all transition processes in the plant itself. Then it is possible to ignore the time delay between the changes of the control signal and the corresponding changes of the plant state and to treat the plant as a static system according to (1).

The movement planner R provides the desired output parameters, i.e., output vector and its derivative. In the simplest case, the distance from the target can be determined by the relation

$$\dot{x}^d = \frac{1}{\tau}(x^a - x^d), \tag{6}$$

where τ is the time constant of the desired changes of the plant state. It is evident that (6) prescribes the exponential decrease of the distance from the target.

The plant is a nonlinear system. Its state is determined by the control signal $c(t)$ according to (1). The control blocks should be also nonlinear structures, but they can be locally linearized. Control functions can be supposed to be linear with respect to the output discrepancy x^Δ according to (2) and to the derivative of the desired output $\dot{x}^d$ according to (3). It results in better convergence of learning process.

3. Feedback block

The feedback block B keeps the system stable and ensures that the plant output will reach the target x^a. It realizes the transfer function (2) and its influence gradually decreases with the

improving function of the neurocontroller. The feedback block B can be implemented in a neural network and adapted by the Hebbian rule. The chosen control signal c is transformed into the plant output x according to (1). Let random and uniformly distributed small disturbances Δc be generated around some point c in the control signal space. These disturbances cause the corresponding small changes of the plant output Δx. The Hebbian learning can proceed according to the equations:

$$B(c) = \sum \Delta B(c), \tag{7}$$

$$\Delta B(c) \sim \Delta c \cdot \Delta x^{\mathrm{T}}. \tag{8}$$

The plant output is described by (1) and its differences approximately by

$$\Delta x \doteq J(c) \cdot \Delta c, \tag{9}$$

where $J(c)$ is the Jacobian of the plant transfer function. Then it can be obtained from (7)–(9):

$$B(c) \sim \sum (\Delta c \cdot \Delta c^{\mathrm{T}} \cdot J^{\mathrm{T}}(c)) \tag{10}$$

and due to the uniform distribution of Δc,

$$B(c) \doteq \beta \cdot J^{\mathrm{T}}(c), \tag{11}$$

where β is the normalizing constant. Hence, the feedback block B learns the transposed Jacobian $J^{\mathrm{T}}(c)$ of the plant transfer function in the designated point. If the disturbances of the control signal Δc are generated in different points of the operational space, then the block B learns the mean value of the transposed Jacobian. It can learn the complex nonlinear function given by (11) within the whole operational space, if it is implemented in some multilayer neural network, e.g., according to Fig. 2 in the next section.

4. Feedforward block

The feedforward block F ensures that the plant output will follow the prescribed trajectory with the prescribed derivative. It realizes the transfer function (3) and its influence increases with the improving function of the neurocontroller. The feedforward block F can be implemented in a three-layer neural network according to Fig. 2. It contains four subnetworks $\mathrm{F_D}$, $\mathrm{F_O}$, $\mathrm{F_L}$, and $\mathrm{F_N}$, with corresponding matrices of synaptical weights.

The hidden layer consists of a set of multiplicative units that perform the elementwise multiplication. Every multiplicative unit involves a linear input neuron h_{L}, a nonlinear input neuron h_{N}, and a multiplicative output neuron h_{M}. The input neurons are activated in the standard way

$$h_{\mathrm{L}} = F_{\mathrm{L}} \cdot \dot{x}^{\mathrm{d}}, \tag{12}$$

$$h_{\mathrm{N}} = \sigma(F_{\mathrm{N}} \cdot c + \theta_{\mathrm{N}}), \tag{13}$$

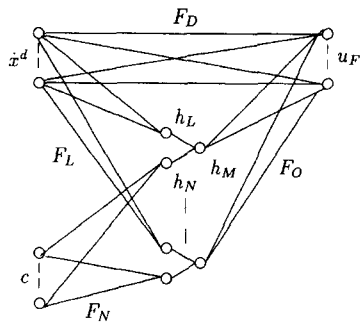

Fig. 2. Implementation of the feedforward block F.

where σ is the sigmoidal function and θ_N is the threshold vector. The output neurons perform the elementwise multiplication of input activations

$$h_M = h_L \otimes h_N . \tag{14}$$

They can be realized by higher-order neurons.

The output control signal consists of two functions:

$$u_F = F_D \cdot \dot{x}^d + F_O \cdot h_M . \tag{15}$$

The first part is independent of the control variable c and improves the convergence of learning. The second part is dependent on the control variable c, reflects the plant nonlinearity, and ensures high accuracy of learning. The feedforward block F can learn according to the error back-propagation procedure [10] that minimizes the output error by steepest descent method. However, the output control error $e_{\dot{c}}$ is not expressed explicitely in the proposed scheme, and it must be somehow estimated. It can be shown [7] that the error of the plant output derivative $e_{\dot{x}}$ is back propagated through the plant according to

$$e_{\dot{c}} = J^T(c) \cdot e_{\dot{x}} , \tag{16}$$

where $e_{\dot{x}} = \dot{x}^d - \dot{x}^c = \dot{x}^\Delta$. Derivative of (2) gives

$$\dot{u}_B \simeq \beta \cdot J^T(c) \cdot \dot{x}^\Delta \tag{17}$$

and, therefore, the output control error can be estimated by

$$ee \sim \dot{u}_B . \tag{18}$$

It is obvious, that learning of the feedforward block F requires the aid of the block B, that must have learned in advance, and that estimates the output control errors. The learning process is performed on the control structure presented in Fig. 1. Targets must be chosen in the working output space of the plant. The control problem is solved through time for sequences of random trials. Matrices of synaptical weights of the subnetworks presented in Fig. 2 are adapted according

to the error back-propagation algorithm. The utility function is chosen as the total square error:

$$U = \sum \| e_{\dot{c}} \|^2 \tag{19}$$

and its minimization according to the steepest descent method requires the changes of individual synaptical weights w_{ij} generally according to the negative gradient

$$g(w_{ij}) \sim \dot{x}_j^{\mathrm{d}} \cdot e_{\dot{c}i}. \tag{20}$$

In the presented scheme, it leads to the following relations:

$$g(F_{\mathrm{D}}) \sim \dot{u}_{\mathrm{B}} \cdot (\dot{x}^{\mathrm{d}})^{\mathrm{T}}, \tag{21}$$

$$g(F_{\mathrm{O}}) \sim \dot{u}_{\mathrm{B}} \cdot (h_{\mathrm{M}})^{\mathrm{T}}, \tag{22}$$

$$g(F_{\mathrm{L}}) \sim (((F_{\mathrm{O}})^{\mathrm{T}} \cdot \dot{u}_{\mathrm{B}}) \otimes h_{\mathrm{N}}) \cdot (\dot{x}^{\mathrm{d}})^{\mathrm{T}}, \tag{23}$$

$$g(F_{\mathrm{N}}) \sim (((F_{\mathrm{O}})^{\mathrm{T}} \cdot \dot{u}_{\mathrm{B}}) \otimes h_{\mathrm{D}}) \cdot (c)^{\mathrm{T}}, \tag{24}$$

$$g(\theta_{\mathrm{N}}) \sim ((F_{\mathrm{O}})^{\mathrm{T}} \cdot \dot{u}_{\mathrm{B}}) \otimes h_{\mathrm{D}}, \tag{25}$$

$$h_{\mathrm{D}} = h_{\mathrm{L}} \otimes h_{\mathrm{N}} \otimes (1 - h_{\mathrm{N}}), \tag{26}$$

where h_{L}, h_{N}, h_{M} are given by (12)–(14). The convergence of the learning process may be improved by using the learning with momentum [10]. It results generally in the relation

$$T_Y \cdot \frac{\mathrm{d}^2 Y}{\mathrm{d}t^2} + \frac{\mathrm{d}Y}{\mathrm{d}t} = \alpha_Y \cdot g(Y), \tag{27}$$

where Y stands for F_{D}, F_{O}, F_{L}, F_{N} and θ_{N}; T_Y and α_Y are the learning parameters for the corresponding subnetwork. The initial states of the matrices F_{D}, F_{O}, F_{L}, F_{N} and the vector θ can be arbitrary, but the matrices F_{L} and F_{N} must not be zeroes.

The operation of the whole system can be described according to (1)–(5) by the differential equation

$$\dot{x}^{\mathrm{c}} + J(c) \cdot B(c) \cdot x^{\mathrm{c}} = J(c) \cdot B(c) \cdot x^{\mathrm{d}} + J(c) \cdot F(c) \cdot \dot{x}^{\mathrm{d}}. \tag{28}$$

As $B(c) \sim J^{\mathrm{T}}(c)$ according to (11), the product $J(c) \cdot B(c)$ is a positive definite matrix and the control is stable.

The convergence of the learning process can be analyzed for a linear case. Let the linear transfer functions of the blocks P, F and B be determined by the matrices P, F and B, respectively. Then $x = P \cdot c$, $J = P$, $B = \beta \cdot P^{\mathrm{T}}$, $u_{\mathrm{F}} = F \cdot \dot{x}^{\mathrm{d}}$ and $u_{\mathrm{B}} = B \cdot x^{\Delta}$, where the constant β specifies the gain of the feedback loop. It is evident that the block F in Fig. 2 is reduced to the linear subnetwork $\mathrm{F_D}$ and its learning is described by (27) for $Y = F_{\mathrm{D}}$. Let us introduce the matrix variables Φ and Ψ according to

$$\Phi = P \cdot F - E, \tag{29}$$

$$\Psi = (x^{\mathrm{d}} - x^{\mathrm{c}}) \cdot (\dot{x}^{\mathrm{d}})^{\mathrm{T}}, \tag{30}$$

where E is the unit matrix. Then (17), (21), (27), (28) and derivative of (30) and (6) give

$$T\cdot\frac{d^2\Phi}{dt^2}+\frac{d\Phi}{dt}+\alpha\cdot\beta\cdot P\cdot P^T\cdot[\beta\cdot P\cdot P^T\cdot\Psi+\Phi\cdot\dot{x}^d\cdot(\dot{x}^d)^T]=0, \tag{31}$$

$$\frac{d\Psi}{dt}+(E/\tau+\beta\cdot P\cdot P^T)\cdot\Psi+\Phi\cdot\dot{x}^d\cdot(\dot{x}^d)^T=0. \tag{32}$$

The variable $\dot{x}^d$ is related to the operational process. It varies significantly during every movement in every trial with the time constant τ. On the other hand, the variable Φ is related to the entire learning process. It changes relatively slowly and should not follow steep variations in individual movements. All time constants of the variable Φ are supposed to be large in comparison with the time constant τ of individual movements. Then statistical dependences between variables $\dot{x}$ and Φ can be neglected. The variable Ψ consists of fast and slow components having, respectively, small and large time constants and being related, respectively, with the variables $\dot{x}$ and Φ.

Let us average (31) and (32) for a limited series of movements and denote the mean values of Φ and Ψ by $\bar{\Phi}$ and $\bar{\Psi}$. As the variables $\dot{x}$ and Φ are supposed to be statistically independent, we can put

$$\overline{\Phi\cdot\dot{x}^d\cdot(\dot{x}^d)^T}=\bar{\Phi}\cdot\overline{\dot{x}^d\cdot(\dot{x}^d)^T}. \tag{33}$$

As the learning trials are generated randomly and uniformly, we can write $\overline{\dot{x}^d\cdot(\dot{x}^d)^T}=v^2\cdot E$, where v^2 is the mean square of the desired output vector derivative for any coordinate. Then (31) can be averaged to

$$T\cdot\frac{d^2\bar{\Phi}}{dt^2}+\frac{d\bar{\Phi}}{dt}+\alpha\cdot\beta\cdot P\cdot P^T\cdot[\beta\cdot P\cdot P^T\cdot\bar{\Psi}+v^2\cdot\bar{\Phi}]=0. \tag{34}$$

When averaging (32) for a series of movements, we can ignore the fast component of Ψ and put

$$(E/\tau+\beta\cdot P\cdot P^T)\cdot\bar{\Psi}+v^2\cdot\bar{\Phi}=0. \tag{35}$$

The characteristic equations of the system (34) and (35) split into a set of characteristic equations for eigenvalues μ according to the relation

$$T\cdot\mu^2+\mu+\alpha\cdot\beta\cdot v^2\cdot\lambda_i/(1+\beta\cdot\tau\cdot\lambda_i)=0, \tag{36}$$

where λ_i is the ith eigenvalue of the matrix product $P\cdot P^T$. It is obvious that the learning process is stable, as the real parts of all eigenvalues μ are negative. However, the validity of (33) requires that $|\mu|\cdot\tau\ll 1$ for all eigenvalues. This can be achieved e.g., by increasing the parameter T, what improves the learning stability, but generally deteriorates the learning rate. Therefore, the proper values of the learning parameters must be chosen as a compromise between the rate of learning and its stability.

Finally, an algorithmic summary is presented to aid in the development of a computer model of the neurocontroller.

(1) The dimensionality of the control signal c and of the output state x are given by the plant (1); working space size is given by the task to be solved.

(2) Teach the feedback block B by the Hebbian rule (7), (8). The learning points should be located throughout the working space, small distributions Δc should be generated uniformly in all coordinates of the control signal c. Choose the normalizing constant $\beta \simeq 1$.

(3) Initialize the feedforward block F. Choose the number of the hidden units h. State the learning constants $T \gg 1$, $\alpha \ll 1$ and define the matrices of synaptical weights as random ones (27). Choose the time step $\Delta t \ll 1$, that reflects the computer digitization of the continuous process.

(4) Teach the feedforward block F by the error back-propagation learning. Generate randomly trajectories inside the working space with the time constant $\tau \simeq 1$ (6). For every time step compute the output error $\Delta \dot{x} = \dot{x}^{d} - \dot{x}^{c}$ and adapt the matrices of synaptical weights according to (21)–(27). Compute the average square root error for several movements. Finish the learning process when the total error is small enough.

5. Numerical experiments

The differential neurocontroller described above was tested by a computer in two numerical experiments. The first one analyzed relations between the rate and stability of the learning process; the second one tested the influence of nonlinearity.

A linear plant with a random transfer matrix with dimensions 3×6 was used in the first experiment. The rate of the learning process was analyzed in dependence on the system parameters presented in (36). It was found that $\| \bar{\Phi} \|$ decreases exponentially with the time of learning if $\eta = |\mu|_{\max} \cdot \tau \ll 1$. The rate of learning, i.e., the exponent of the decrease of $\| \bar{\Phi} \|$, is approximately equal to the minimal eigenvalue $|\mu|_{\min}$ of the system (36). The rate of learning increases proportionally to increasing $|\mu|_{\min}$. However, increase of $|\mu|_{\min}$ generally results in increase of η, what causes discrepancy between $|\mu|_{\min}$ and the rate of learning. This discrepancy increases until $\eta \simeq 1$. Then the convergence of the learning process fails.

The second experiment treated with a two-links nonlinear model of the arm with six degrees of freedom in three-dimensional space. Eight multiplicative units were used in the hidden layer of the block F. It was found that for the small size of the operation space, i.e., when the plant can be approximated by a linear system, the convergence of the learning process can be described in the same way as in the linear case. When the size of the operation space increases, the nonlinear subnetworks must be taken into account and all relations are rather complicated. Nevertheless, a high rate and accuracy of learning may be gained if the parameters of the neural nets are were well tuned. The convergence depends on the form of the transfer function $F(c)$ of the feedforward block F. If only the linear function is used (i.e., $F_{O} = 0$), then the convergence is good, but the error stays relatively high. If only the nonlinear function is used (i.e., $F_{D} = 0$), then the convergence of the learning process may be slower and sometimes even nonmonotonic. The use of both the linear and nonlinear functions gives a good convergence and small errors.

6. Conclusion

The proposed structure of the differential neurocontroller has several advantages. Control by differences causes the architecture to be relatively simple. The plant is locally linearized, which

results in a good convergence of the learning process. The developed procedure of the design does not require any calculation of a matrix inversion. The transfer function of the plant need not be explicitly known and learning can be performed by using the real plant.

The plant has been treated as a static system that suits to large time constants of desired output changes. If extremely fast output changes are required, derivatives of the plant state (or previous states) must be taken into account. This problem is to be solved in the next work.

References

[1] R. Caminiti, P.B. Johnson, C. Galli, S. Ferraina, Y. Burnod and A. Urbano, Making arm movements within different parts of space: The premotor and motor cortical representation of a coordinate system for reaching to visual targets, *J. Neuroscience* **10** (1991) 2039–2058.

[2] K.S. Fu, R.C. Gonzalez and C.S.G. Lee, *Robotics: Control, Sensing, Vision and Intelligence* (McGraw-Hill, New York, 1987).

[3] A.P. Georgopoulos, A.B. Schwartz and R.E. Kettner, Neural populatio coding the motor direction, *Science* **233** (1986) 1357–1440.

[4] M. Ito, *The Cerebellum and Neural Control* (Raven Press, New York, 1984).

[5] M. Kawato and H. Gomi, A computational model of four regions of the cerebellum based on feedback-error learning, *Biol. Cybern.* **68** (1992) 95–103.

[6] M. Kuperstein and J. Rubinstein, Implementation of an adaptive neural controller for sensory–motor coordination, *Proc. IJCNN Washington* (1989) II-305–310.

[7] D. Psaltis, A. Sideris and A. Yamamura, Neural controllers, *IEEE Internat. Conf. on Neural Networks* **4** (1987) 551–557.

[8] S. Řízek and A. Frolov, Differential control by neural networks, *Neural Network World* **4**(4) (1994) 493–508.

[9] P.J. Werbos, Backpropagation and neurocontrol: A review and prospectus, *Proc. IJCNN Washington* (1989) I-209–216.

[10] P.J. Werbos, Backpropagation through time: What is does and how to do it, *Proc. IEEE* **78** (1990) 1550–1560.